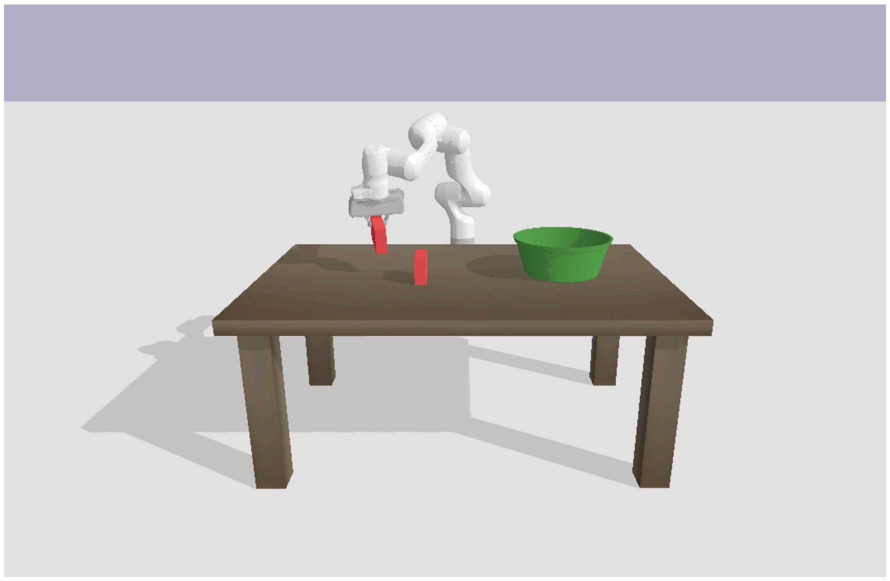
ROB-UY 2004: Robotic Manipulation and Locomotion
Project Report
Sage Cronen-Townsend



Introduction

As an interesting conclusion to the Robotic Manipulation and Locomotion class, we were given a robot arm and a simple manipulation task: picking up two red blocks and placing them in a bowl. While this is relatively easy for most humans to do, it is much more complex when using a robot arm. For one thing, the robot used in this project was the Franka Emika Panda robot, a robot arm with seven revolute joints. In previous labs we have learned to control the motion of a robot arm with three joints, so this marked a departure from what we had explicitly done in the past, and required the application of our learning to a somewhat different problem. In addition, we were challenged to decide which method of control would be most effective in completing the Panda robot's task. There were several additional requirements to our task: that we use no external libraries besides numpy and scipy, that at least one controller act in end effector space, that we must generate smooth motions, and that we must compensate for gravity.

Control Methods

Over the course of this semester we have learned to control the movement of a robot through various techniques. In labs we have implemented each different controller on a robot arm with three degrees of freedom (DoF), while the 7 DoF Panda robot is more complex. However, many of these controllers are generalizable and can be implemented even with this higher level of complexity, especially as we were additionally given a file of helper functions. Considering the requirement that at least one controller work in end effector space, the most reasonable options seemed to be impedance control and resolved rate control.

As the most recent controller we have implemented, impedance control was the first I considered for this task. Impedance control can be used to directly control forces in the end effector, and is useful as it does not require the inverse of the Jacobian matrix. As such, we don't need to be concerned with singularities. Downsides are generally lower position tracking accuracy and the loss of ability to control redundancy, though it was difficult without experience to know if these would prove to be relevant issues. As this task did seem to require good position tracking and we had not yet fully implemented this controller in lab, I decided to proceed with a resolved rate controller.

Throughout the course of the class, many of the labs built up our skills (and functions) which allowed us to implement a resolved rate controller. Resolved rate first calculates a smooth reference trajectory in end effector space, then P control is done in end effector space as well. The inverse Jacobian (or pseudo inverse) then finds the desired joint velocities, and D control is performed in joint space. Though we do have to consider cases of singularities, resolved rate control grants good position tracking accuracy and we have the opportunity to add tasks in the null space. Considering that the manipulation task involved precise movements but was relatively robust to differences in force, I decided to continue with resolved rate control.

Implementation

The steps followed to implement a resolved rate controller for the 7 DoF panda robot were only slightly more complex than with the 3 DoF NYU Finger. To implement the controller we needed to be able to find end effector pose, calculate desired positions and velocities for each time step, and relate joint velocities to end effector twists. Likely the most challenging part of this task was in creating forward kinematics and compute Jacobian functions to achieve these goals, but those were given in the helper functions file. The compute trajectories function could also be lifted from lab 4, as trajectories in end effector space are no more complicated for a 7 DoF robot than 3. After that, the steps to follow were largely the same as in labs 5 and 6, though I did not attempt to avoid singularities as the robot never approached that configuration.

Before starting the simulation I created arrays to store relevant information for plotting, such as measured and desired end effector position. I also used this space to determine the P and D constants, which had to be tuned to allow for the most accurate tracking, and specify the goal positions I wanted to reach. It's worth noting that the positions of the bowl and blocks appear to have been inaccurate in the z dimension, and had to be experimentally adjusted so the robot arm would aim low enough.

Upon entering the main control loop, the simulation was started. Within this loop I used logic to determine the start and goal positions, and whether the gripper should be open or closed. Implementing the resolved rate controller required previous computation of the forward kinematics, oriented Jacobian, and desired position and velocity at each time step. Luckily, all the functions required were either provided or had been previously created in other labs. From there, the actual controller calculated desired end effector velocity using the reference velocity, P, and error in reference and measured positions. The pseudo inverse of the Jacobian (pseudo because the linear velocity section of the Jacobian was not square or invertible) was then used to convert from desired end effector velocity to joint velocities. The torque to send to each joint, then, was calculated as D times the error between desired and measured joint velocities, plus a term which compensated for the effects of gravity at each joint. Gravity compensation was incredibly easy to do as it only involved calculating the effects of gravity using a helper function and adding that term directly to the torques to send, but it proved critical. When I ran the simulation without it, the robot behaved erratically and threw everything off the table.

Results and Analysis

Once debugging was complete, it was not difficult to adjust the path and gripper to successfully reach each block, pick it up, and drop it in the bowl. It was necessary to first aim for a spot above each block before lowering to grab it and ascending again, because otherwise the first block would hit the second and knock it over on its way to the bowl. Alternative fixes could have included dropping each block from farther above the bowl, but it felt prudent to create a more complex path rather than risk outside interference such as wind which might exist outside the simulation. I did some slight readjustment of the goal positions once again so the gripper would hold slightly lower on each block, which did a lot to improve the stability of the grasp. The final design choice I made was in the time spent between each goal position. Because I had such a small distance to travel between the goal above each block and the block itself, it did not make sense to have a consistent time to travel to each goal position. Instead, I used the distance formula so that the time to each goal would be proportional to the distance between start and finish. While the end effector still visibly decelerated when reaching each goal, this is useful for controlling oscillation and swing of the blocks, and overall the trajectory became much more smooth.

Figure 1 below shows the measured and desired x, y, and z positions of the end effector over the course of the simulation. The position tracking is nearly perfect, while the measured velocity shows small deviations from the desired. To ameliorate this effect I tuned the D values, which led to fewer oscillations when switching directions. Considering the complexity of the path, deviations are small, and the task was able to be performed consistently with no issues.

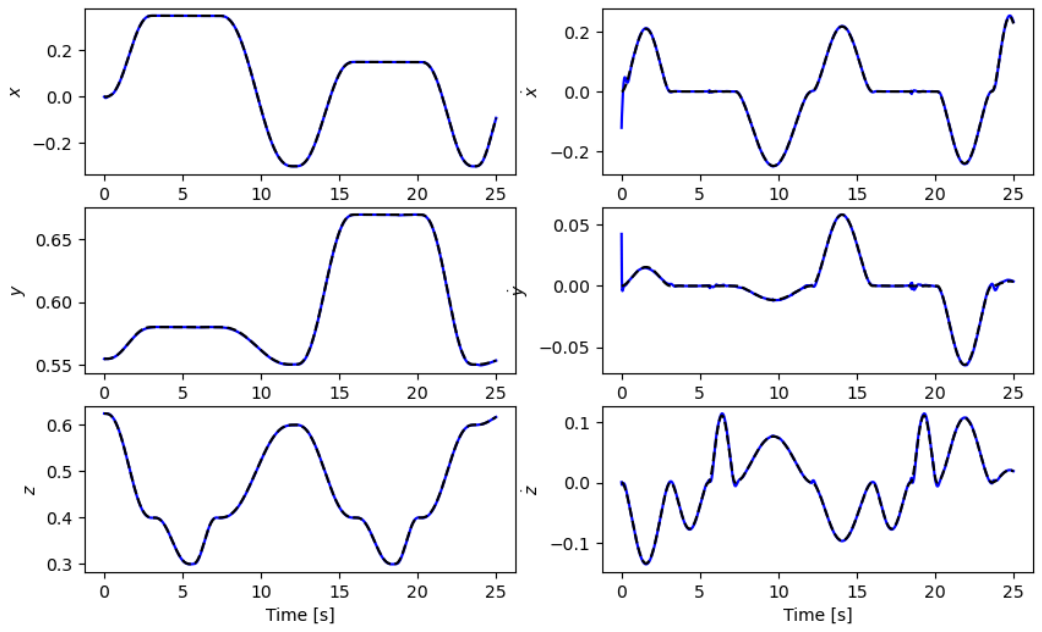


Figure 1: Superimposed measured and desired positions (left) and velocities (right) of the end effector in x, y, and z directions

A graph of x-z position of the end effector is also shown in figure 2. It is helpful for visualizing the goal positions and the path connecting them. It is more clear in this image that the paths between each goal were straight, as determined by the compute trajectory function in end effector space.

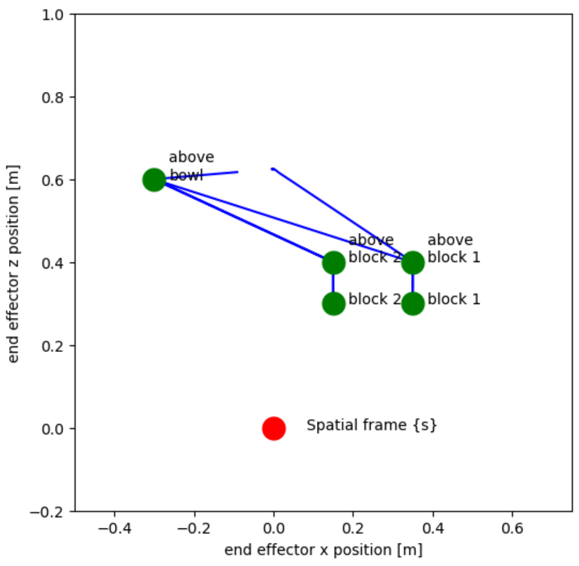


Figure 2: Path of the end effector in x-z coordinates

One effect of resolved rate control is that desired joint positions are never directly specified. In previous labs with the 3 DoF NYU Finger we have used analytic inverse kinematics to find desired joint positions from desired end effector position, although it was not used for control. This was doable because the relative simplicity of the robot ensured there would never be more than two potential configurations for a given end effector position. This is not the case with the 7 DoF Panda robot and inverse kinematics cannot be done. Out of interest I chose to plot the measured joint positions despite not having a desired reference, but the graph of measured and desired joint velocities gives more useful information by far. Both are shown in figure 3 below.

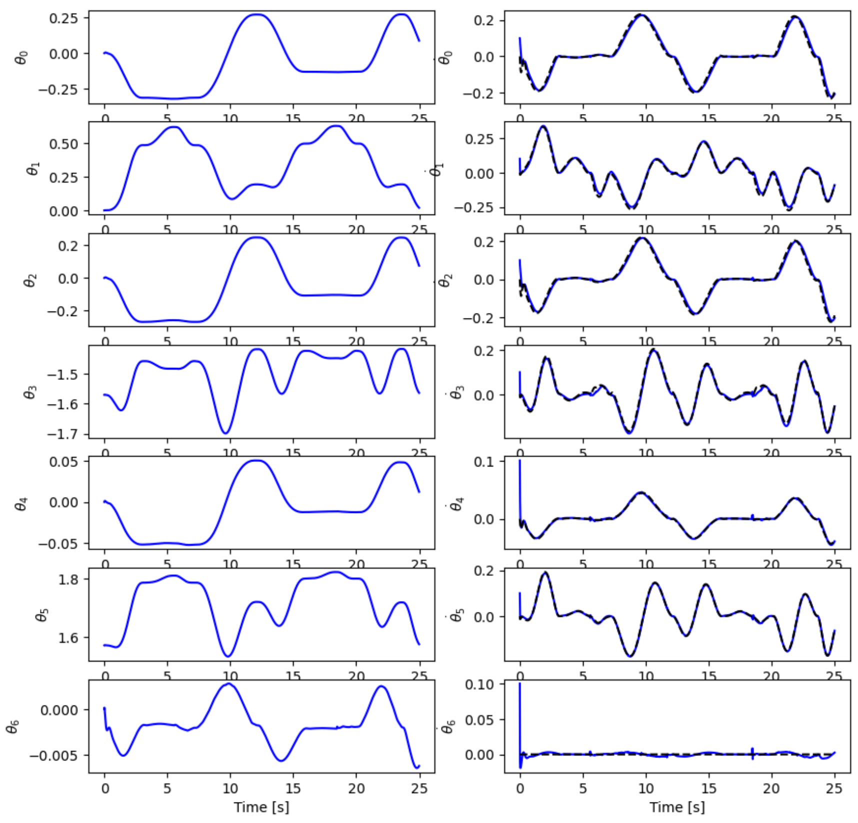


Figure 3: Measured joint positions (left) and measured and desired joint velocities (right)

Though there are no reference positions to compare to, it is interesting to note that all joint trajectories are smooth curves with few, if any, spikes. The joint velocities are also fairly consistent, though each joint shows a spike in velocity at the very beginning of the simulation when the robot starts moving. By tuning the D values I was able to create more accurate tracking, but there is still some deviation from desired velocities. The last joint, specifically, may have been affected by grasping the blocks, as the most noticeable spikes occur when the blocks were picked up and dropped.

Overall, the robot behaves nearly exactly as expected, with all deviations easily explained by either the effects of the gripper and blocks or mistuning of the P and D vectors. While this project was challenging, it was also incredibly rewarding to see the generalization of our knowledge be applicable to a more complex task, and to be able to truly analyze and understand such a system.